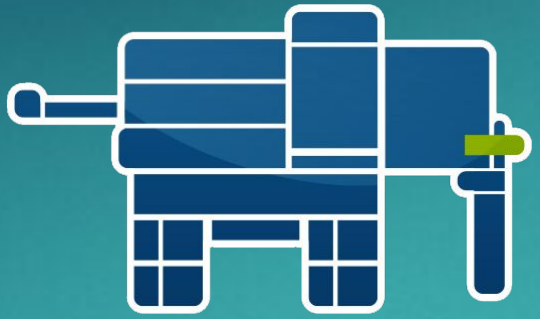
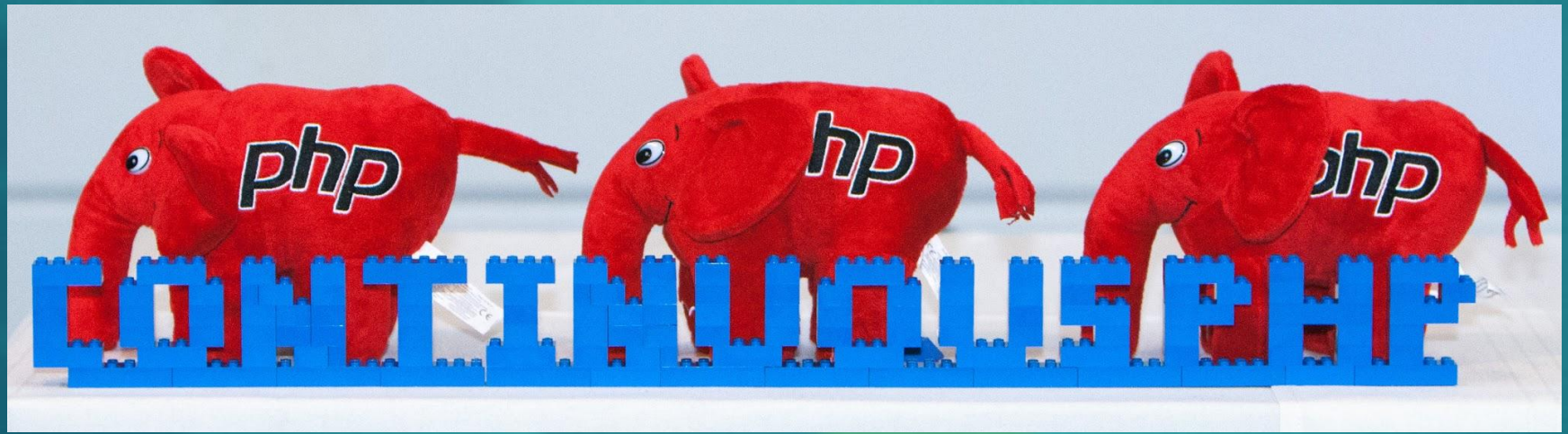


The Continuous Talk



About myself: *Pascal Paulis*

- Co-Founder and Core developer of continuousphp©
- 8+ SysAdmin/PHP consultant background
- PHP & ZF Certified Engineer
- Continuous Delivery/Deployment Expert

https://lu.linkedin.com/pub/pascal-paulis/40/115/ba0

Continuous what?

Continuous Integration (CI)

- continuously merging developer branches in a common one
- build and test every commit to prevent integration issues
- deploy every build on an integration server

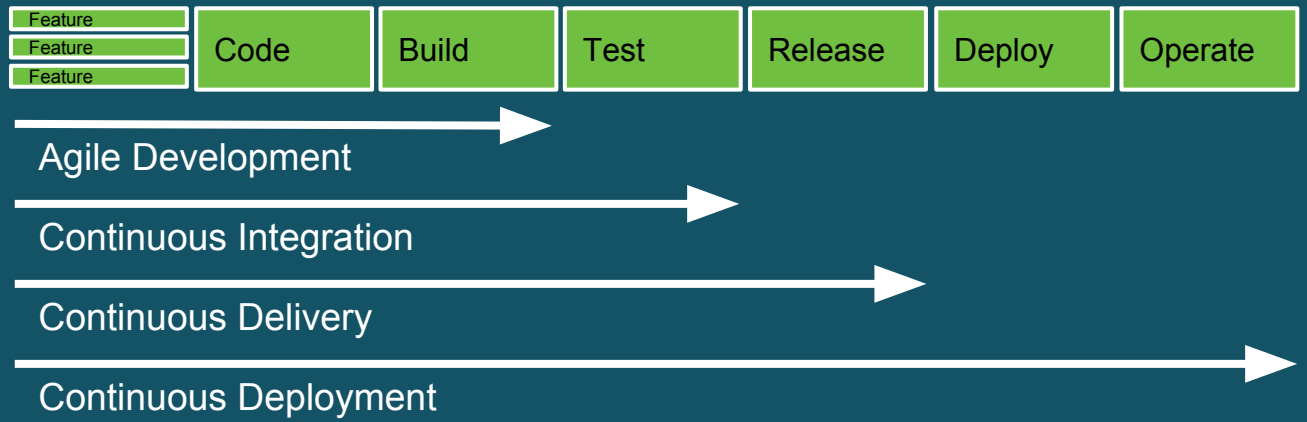
Continuous Delivery (CD)

- ***Continuous Integration*** is a part of CD
- Code is ***packaged*** by a build server every time a change is ***committed***
- any ***code commit*** may be ***released*** to customers at any point
- implements ***Scrum*** Project Management

Continuous Deployment

- ***Continuous Delivery*** is a part of ***Continuous Deployment***
- every successful build is deployed to a Production Environment
- any completed, working feature is delivered to production as soon as possible
- implements ***Kanban*** Project Management

Comparison

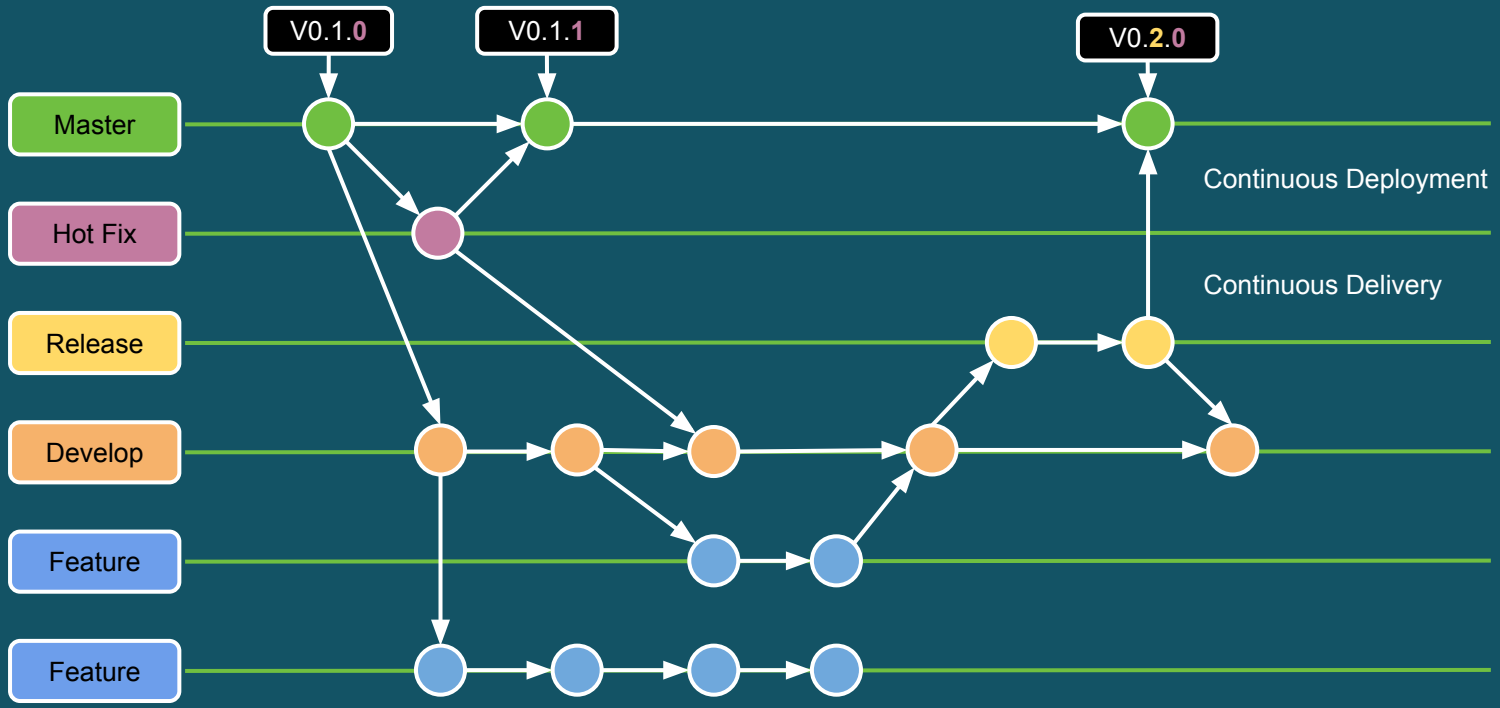


Library development workflow

- Continuous Integration & Continuous Delivery only!
  - need to integrate a new version of the library in the app using it, to trigger the application build
- can trigger client application update on new release
  - update client application's dependency manager
  - trigger a new build of the application

How to organize the daily work ?

Branching model : Git Flow



http://nvie.com/posts/a-successful-git-branching-model/

The Git-Flow tool

- Install:
  - Linux: *sudo apt-get install git-flow* (on Debian/Ubuntu)
  - Mac: *brew install git-flow*

```
cd /my/project/root
git flow init
git flow <feature|hotfix|release> <start|finish|publish> <version#|feature name>
```

  - Git-Flow Tutorial: http://danielkummer.github.io/git-flow-cheatsheet/

Semantic Versioning

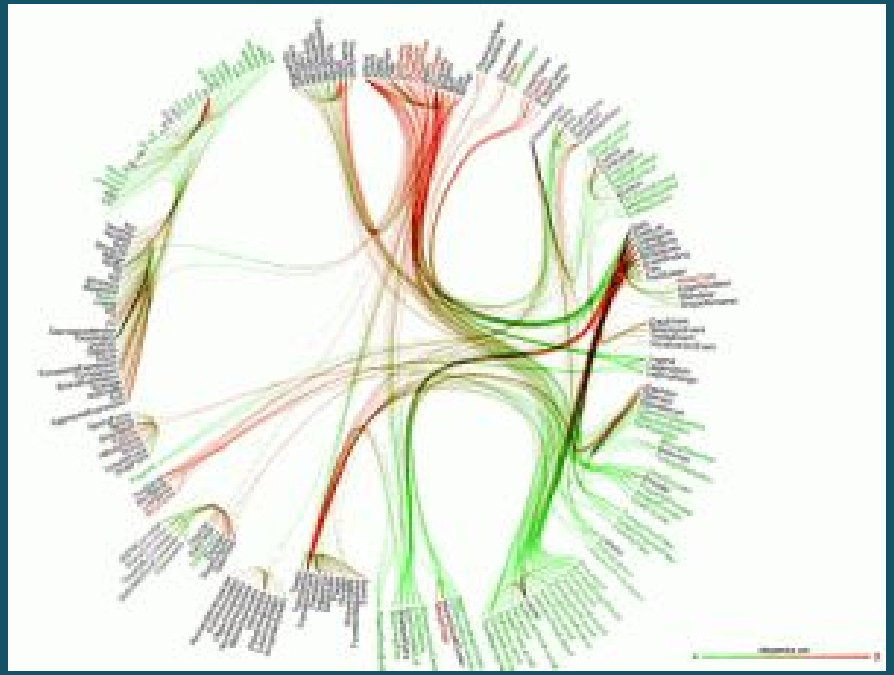
- naming convention to use for Continuous Deployment & Continuous Delivery
  - <MAJOR>.<MINOR>.<PATCH>
    - MAJOR version for backward compatibility breaks
    - MINOR version for backwards-compatible features
    - PATCH version for backwards-compatible bug fixes

How to manage our builds ?

The build goals

- prepare the application for use in production
- … and for testing purposes
- … even for development

“Sounds nice, but our project is too specific!”



No, it’s all about provisioning and dependencies!

System dependencies

- database
- file stores
- web services
- cache
- ...

Code dependencies

- configuration files
- third party libraries

How to manage them?

Composer

- Dependency Manager for PHP
- focused on library dependencies
- define php version and extension requirements
- https://getcomposer.org

Phing

- PHing Is Not GNU make
- PHP Project build tool
- based on Apache Ant
- written in PHP; easily extensible by PHP developers
- define sequences of tasks
- organized in targets

Database migration tools

- Doctrine Migrations
  - Magic with Doctrine ORM
  - Compatible only with MySQL
- Phinx
- dbdeploy
- Or simply use nosql databases :)

I’m not confident enough with my code :(

How do you test your code ?

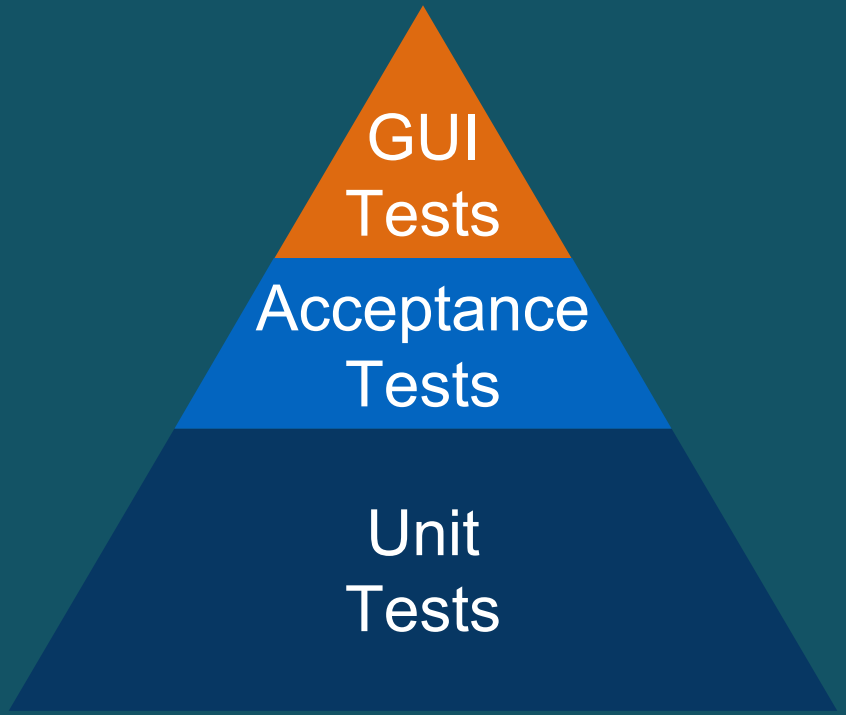
Acceptance Testing with Behat

- Inspired by Cucumber
- Uses Gherkin syntax to define specs
- Supports several web browser drivers through extensions

Unit Testing Tools

- PHPUnit
  - de facto industry standard
  - created in 2001
- phpspec
  - tests designed by specification
  - eases TDD approach
- atoum
  - simple and intuitive
  - Good performances

Test pyramid



Continuous Deployment… takes also care about deployment!

Deployment Tools

- Amazon CodeDeploy
  - Used by amazon.com for years
  - Support deploy on boot
  - Support rollback
  - Support transitional deploy
  - Free on AWS
- Zend Deployment
  - Included in Zend Server
  - Support rollback
  - Support transitional deploy

Thank you!


https://www.google.com/+Continuousphp


https://www.linkedin.com/company/continuousphp


@continuousphp


https://www.facebook.com/pages/ContinuousPHP/1830472518485 48

continuousphp.com